CSE 2017 Data Structure and Lab

Final Exam, Spring 2019

Student Number: _____

Name: _____

Part 1. True/False Question [2 points each].

Decide whether this sentence is True or False. You should also give brief reasons for your answers.

1.1 **True** **False** A tree is an example of a linear data structure

1.2 **True** **False** If arity of operators is fixed, then both postfix notation and prefix notation can be used to parse expressions without parentheses?

1.3 **True** **False** In a complete k-ary tree, every internal node has exactly k children or no child. The number of leaves in such a tree with n internal nodes is (n-1)*k.

1.4 **True** **False** The height of a binary tree is the maximum number of edges in any root to leaf path. The maximum number of nodes in a binary tree of height h is 2^h-1.

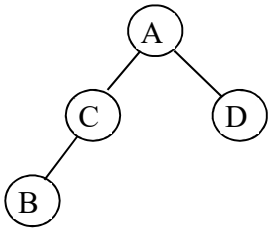
1.5 **True** **False** A full binary tree with n leaves contains 2^n.

1.6 **True** **False** Consider a binary max-heap implemented using an array. Array 25,14,16,13,10,8,12 represents a binary max-heap

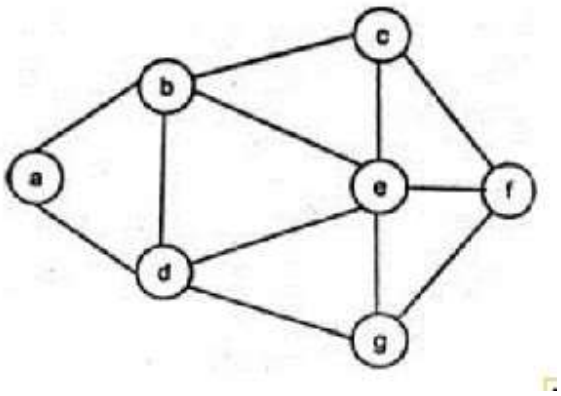
1.7 **True** **False** The content of the array after two delete operations on the correct answer to the previous question is 14,13,8,12,10.

1.8 **True** **False** *** +a −bc − /de − +fgh** is a prefix form of the following expression (a + (b − c))* ((d − e)/(f + g − h)).

1.9 **True** **False** The following binary trees has its inorder and preorder traversals as BCAD and ABCD, respectively



1.10 **True** **False** i) a b e f c g d
ii) a d g e b c f
iii) a d b c g e f.
All of the above is (are) possible output sequence of nodes of a Depth First Search (DFS) starting at node a.



1.11 **True** **False** DFS and BSF can be done in O(V + E) time for adjacency list representation. These operations take $O(V^2)$ time in adjacency matrix representation. Here is V and E are number of vertices and edges respectively.

1.12 **True** **False** The maximum number of edges in an acyclic undirected graph with n vertices is 2*n - 1.

Part 2. Short Answer Questions.

2.1 What is the maximum number of binary trees that can be formed with three unlabeled nodes? Draw all passible binary trees[5 points].

2.2 A scheme for storing binary trees in an array X is as follows. Indexing of X starts at 1 instead of 0. the root is stored at X[1]. For a node stored at X[i], the left child, if any, is stored in X[2i] and the right child, if any, in X[2i+1]. What is the minimum size of X to be able to store any binary tree on n vertices[5 points].

2.3 The height of a tree is the length of the longest root-to-leaf path in it. The maximum and minimum number of nodes in a binary tree of height 5 are [5 points].

2.4 The in-order and pre-order traversal of a binary tree are d b e a f c g and a b d e c f g respectively. The post order traversal of a binary tree is [5 points].

2.5 What does the following function do for a given binary tree? [5 points each].

```
int fun(struct node *root)
{
  if (root == NULL)
     return 0;
  if (root->left == NULL && root->right == NULL)
     return 0;
  return 1 + fun(root->left) + fun(root->right);
}
```

Part 3. Coding & long answers

3.1 Show the expression tree of the equation 2 – 5 / 5 – (4 – 2 – 1). Also, write the result of post order traversal of this expression. Furthermore, when calculating this postfix notation using a stack, Show the change of the stack[10 points].

3.2 Write a C++ program for three methods of minimum heap data structure class.
right(int parent) returns an index value of right child of parent node,
parent(int child) returns an index value of parent of child node,
deleteMin() is a delete function for minimum heap[10 points].

```
#include <iostream>
#include <cstdlib>
#include <vector>
#include <iterator>
using namespace std;
class BinaryHeap
{
```

```
    private:
        vector <int> heap;
        int left(int parent);
        int right(int parent);
        int parent(int child);
        void heapifyup(int index);
        void heapifydown(int index);
    public:
        BinaryHeap()
        {}
        void Insert(int element);
        void DeleteMin();
        int ExtractMin();
        void DisplayHeap();
        int Size();
};


void BinaryHeap::DeleteMin()
{
        [redacted]
[redacted]
[redacted]
}

int BinaryHeap::right(int parent)
{
    [redacted]
[redacted]
}
```

```
/*
 * Return Parent
 */
int BinaryHeap::parent(int child)
{

}
```

3.3 Consider the function printKth() receives root of a Binary Search Tree (BST), a positive integer k as arguments and write Kth largest value in BST with recursion. [10 points].

```
typedef struct Node {
        int data;
        struct node *left;
        struct node *right;
}Node;

class BST  {
 public:
        Node *root;
        int count = 0;
        void create();

        void PrintKth(TreeNode &root, int k);

 };

        void BST::printKth(struct node *root, int k)
        {




        }
```

3.4 Write a C++ operation named maxDepth() that function is supposed to calculate the maximum depth or height of a Binary tree -- the number of nodes along the longest path from the root node down to the farthest leaf node. You should use a recursion[10 points].

```
typedef struct Node {
        int data;
        struct node *left;
        struct node *right;
}Node;

class binary_tree  {
 public:
        Node *root;
        void create();
        void insert(Node *root, node *New);
        void delete(Node *root);
        maxDepth(struct node* node);

 };


int binary_tree ::maxDepth(struct node* node) {

 }
```

3.5 The function for depth first search of graph nodes is shown below. Fill in the blanks[10 points].

```
template <class VertexType>
void DepthFirstSearch(GraphType<VertexType> graph,
VertexType startVertex, VertexType endVertex) {
      StackType<VertexType> stack;
      QueType<VertexType> vertexQ;
      bool found = false;
      VertexType vertex;
      VertexType item;
      graph.ClearMarks();
      [                              ];
      do {
            [                              ];
            if(vertex == endVertex)
                  found = true;
            else
                  if(!graph.IsMarked(vertex)) {
                        [                              ];
                        graph.GetToVertices(vertex, vertexQ);
                        while(!vertexQ.IsEmpty()) {
                              [                              ];
                              if(!graph.IsMarked(item))
                                    [                              ];
                        }
                  }
      } while(!stack.IsEmpty() && !found);
      if(!found)
      cout << "Path not found" << endl;
}
```

Part 4. Essay Writing(Before and After)

4.1 Write your feeling and thinking about yourself before and after taking this course[5 points] What is improvement of your skills or attitude?